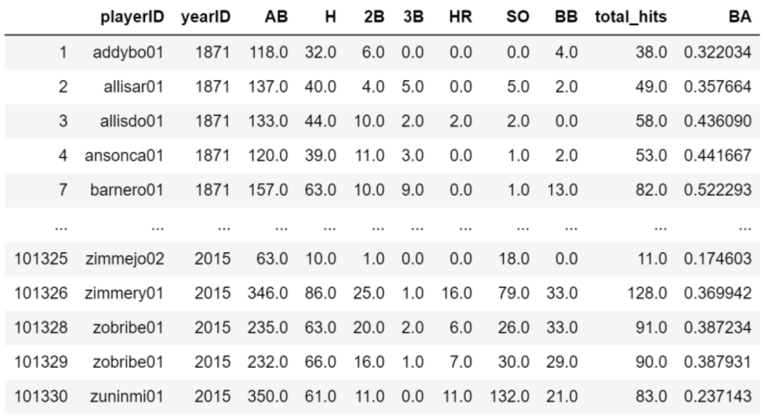
Capstone 2 Baseball Report

By Thomas McMahon

Over the past decade sports have become much more data driven, none more than baseball. In sports to gain the upper hand over your opponents you either need to have deeper pockets than them, or you can be smarter than they are. This second idea is the option that the Oakland Athletics needed to choose in order to compete with bigger market teams such as the New York Yankees. How they accomplished that is through the use of data. Former professional baseball player turned manager Billy Bean after losing another star player to money decided to change the way a team could be put together. With the help on analysts, he determined that he was going to put the cheapest team on the field with the highest on base percentage. This is the idea that inspired this idea to be able to predict how a player well a player is going to bat in future seasons. This project will be able to be used by anybody that is trying to put gain a competitive advantage over opponents in fantasy baseball or even by MLB teams trying to decide which way to move forward with there franchises when it comes to who to sign and who to let go. This project will take public information on all players stats going back to 1970.

The first part of the project is to take a look at what data is available, and when it comes to baseball there was more than enough data when it comes to batting. Although when I first looked at the data the players actual batting average wasn't one of the features in the data. Lucky for me the features did include the number of at bats each player had in each season, as well as the total number of each type of hit (single, double, triple, and home run), so it was easy to produce the total number of hits each player had during the season, and the batting average. After having all of the data clean and having the actual feature that we wanted to predict it was time to start looking at trends in the data using exploratory data analysis.

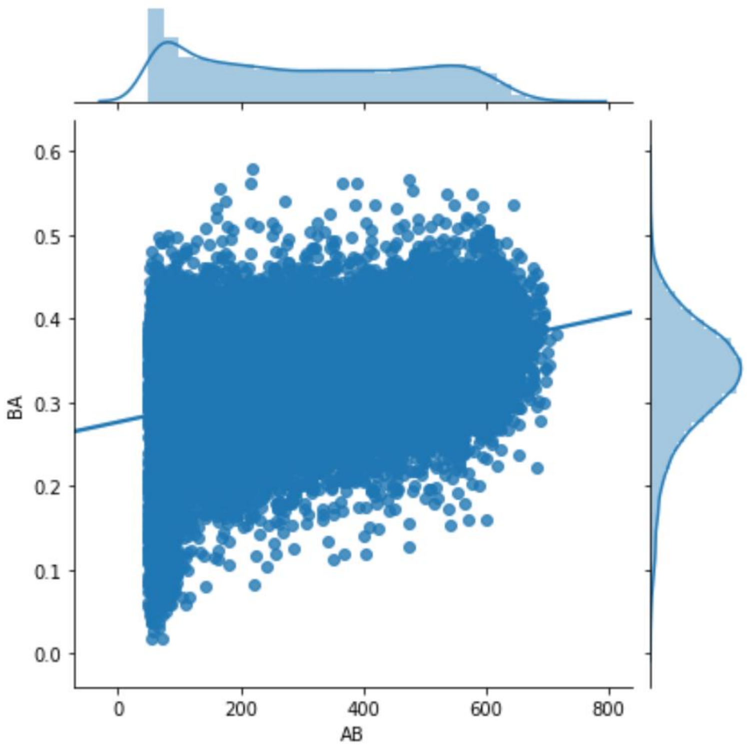


	playerID	yearID	AB	H	2B	3B	HR	SO	BB	total_hits	BA
1	addybo01	1871	118.0	32.0	6.0	0.0	0.0	0.0	4.0	38.0	0.322034
2	allisar01	1871	137.0	40.0	4.0	5.0	0.0	5.0	2.0	49.0	0.357664
3	allisdo01	1871	133.0	44.0	10.0	2.0	2.0	2.0	0.0	58.0	0.436090
4	ansonca01	1871	120.0	39.0	11.0	3.0	0.0	1.0	2.0	53.0	0.441667
7	barnero01	1871	157.0	63.0	10.0	9.0	0.0	1.0	13.0	82.0	0.522293
...	...	...	...	...	...	...	...	...	...	...	...
101325	zimmejo02	2015	63.0	10.0	1.0	0.0	0.0	18.0	0.0	11.0	0.174603
101326	zimmery01	2015	346.0	86.0	25.0	1.0	16.0	79.0	33.0	128.0	0.369942
101328	zobribe01	2015	235.0	63.0	20.0	2.0	6.0	26.0	33.0	91.0	0.387234
101329	zobribe01	2015	232.0	66.0	16.0	1.0	7.0	30.0	29.0	90.0	0.387931
101330	zuninmi01	2015	350.0	61.0	11.0	0.0	11.0	132.0	21.0	83.0	0.237143

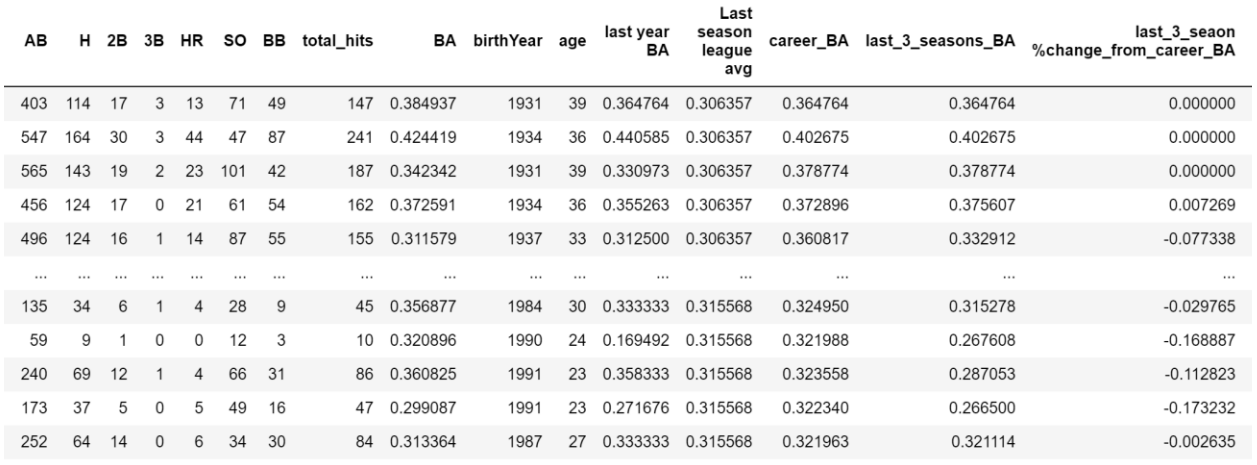
One of the things that I noticed right away when looking at the trends in the columns that were in the data were that as expected as the total number of hits increased in general so did the batting average. But a problem that I quickly saw with that relationship was how wide of a range there was in at bats and batting average when it came to players with very few at bats. This makes sense because the data have every player with at least one at bat so if that one at bat was a hit it registered as the player having a batting average as 100% and the opposite was true with the player have a batting average as 0%. This lead to the decision of removing any player that had less than 50 at bats during a season, this was a judgment call on the number of at bats. I chose 50 because while it still had the widest spread in batting average by far, I didn't want to remove too much data that could change the predictions of the models I would be building. I did find that most of the features that were in my dataset had a positive correlation with batting average, so I knew that I was going to need to come up with some new features to help the model make its predictions.

```python
sns.jointplot(x='AB',y='BA',data=data,kind='reg')
```

```
<seaborn.axisgrid.JointGrid at 0x1a48e316ec8>
```



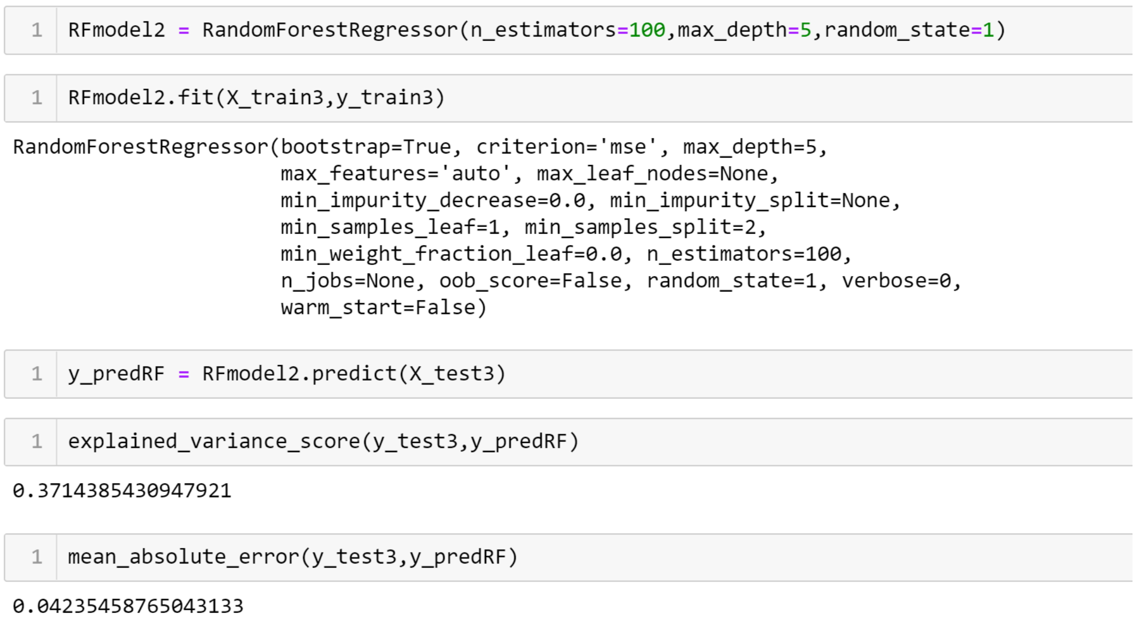
When all of the features having a positive correlation with batting average I knew that feature engineering was going to be a very important part of this project. I was trying to find other features that could be useful and I remembered a lecture that I watched before starting this project and it talked about being able to predict a serries of players batting averages using regression to the mean. What he was doing was using a players past performance as well as the average of all players and saying that a player might have a good or bad season, but they are more likely in the future to have an average that is closer to the average of all players with the assumption that all players are roughly as skilled as one another, so I decided to add a feature of the leagues batting average. Other features that I knew I wanted to add to the data were the players career batting average, the players previous season average, as well as a 3 year trend to see if the player had been having statistically good or bad seasons. After coming up with these new features I realized the way I had the data set up that I would end up cheating on the predictions. This was because I was included the years average I was trying to predict in most of the other features as well as using the current years stats on hits to predict, which wouldn't work. The way that I was able to correct this was by shifting all of the features back a year while leaving the batting average in the current year. The result of this produced a dataset that for each row all of the data other than the batting average only took into account that from previous seasons.



AB	H	2B	3B	HR	SO	BB	total_hits	BA	birthYear	age	last year BA	Last season league avg	career_BA	last_3_seasons_BA	last_3_seaon %change_from_career_BA
403	114	17	3	13	71	49	147	0.384937	1931	39	0.364764	0.306357	0.364764	0.364764	0.000000
547	164	30	3	44	47	87	241	0.424419	1934	36	0.440585	0.306357	0.402675	0.402675	0.000000
565	143	19	2	23	101	42	187	0.342342	1931	39	0.330973	0.306357	0.378774	0.378774	0.000000
456	124	17	0	21	61	54	162	0.372591	1934	36	0.355263	0.306357	0.372896	0.375607	0.007269
496	124	16	1	14	87	55	155	0.311579	1937	33	0.312500	0.306357	0.360817	0.332912	-0.077338
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
135	34	6	1	4	28	9	45	0.356877	1984	30	0.333333	0.315568	0.324950	0.315278	-0.029765
59	9	1	0	0	12	3	10	0.320896	1990	24	0.169492	0.315568	0.321988	0.267608	-0.168887
240	69	12	1	4	66	31	86	0.360825	1991	23	0.358333	0.315568	0.323558	0.287053	-0.112823
173	37	5	0	5	49	16	47	0.299087	1991	23	0.271676	0.315568	0.322340	0.266500	-0.173232
252	64	14	0	6	34	30	84	0.313364	1987	27	0.333333	0.315568	0.321963	0.321114	-0.002635

With all of the features the way that I wanted them it was time to model the data. To do this the first thing that was required was to brake up the data into all the features that I wanted to use to predict the Batting average, and nothing but the batting averages. After this I needed to make sure to scale the data to make sure the results of the model would not be skewed because of the difference in magnitude or standard deviation between features. This process transforms all of the means and standard deviations and makes them all the same. After this it was time to train and test the model. In my first model I used train test split to randomly select the training, and test data. From here I fit a linear regression model, and used it to make predictions to compare to the actual testing data. The result wasn't very promising, but it was a start. I found that the model had a mean error of .043 which is a start, but leaves a pretty wide range statistically on the predictions. My first though was that maybe the model just picked bad data points to train and test on so in my next model I controlled for that. The way that I did that was by selecting all of the data from before 2005 as the training data and everything from 2005 and after as testing data. Using the same linear regression with the new training and test datasets I found that this model actually preformed ever so slightly worse than the first model. After this I figured I needed a different type of model.

For my last two models I decided on a random forest, because this would make multiple models and learn from each one to make better predictions. The was that this is done is using multiple individual decision trees, after each iteration of a tree the model learns what worked and what didn't then runs the next tree. Even with the new improvements the better of the 2 random forests that I created only performed .001 better than its linear regression counterpart.



```
RFmodel2 = RandomForestRegressor(n_estimators=100,max_depth=5,random_state=1)
```

```
RFmodel2.fit(X_train3,y_train3)
```

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=5,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=100,
                      n_jobs=None, oob_score=False, random_state=1, verbose=0,
                      warm_start=False)
```

```
y_predRF = RFmodel2.predict(X_test3)
```

```
explained_variance_score(y_test3,y_predRF)
```

```
0.3714385430947921
```

```
mean_absolute_error(y_test3,y_predRF)
```

```
0.04235458765043133
```

In conclusion although the model did not preform as well as I would have liked, before Billy Beans so called crazy money ball experiment this model would be more effective than how baseball scouts used to try to predict how a player was going to play. Some of these old metrics were how hot the girlfriend or wife of the player was with the rational being if they are hot then the player is more confident in themselves and therefore would preform better on the field. There may have been some features that I could have overlooked or data that I didn't have, such as could the precent of righthanded vs lefthanded pitchers change the batting average, because statistically teams defensively want to have righthanded pitchers throwing to righthanded batters and vice versa. This would have been something that I would have liked to add to the model but I don't have the information on what hand the players bat with so even knowing what hand the pitchers throw with it wouldn't matter because I don't have that information on the batters. I defiantly think this is a worth wild project although maybe not with the information we have now, but maybe down the line with more stats being tracked I wouldn't be surprised if in a near future teams will be able to more accurately predict how a player will preform.